(19)日本国特許庁（ＪＰ）　(12)公表特許公報（Ａ）　(11)特許出願公表番号
特表2001－508908
（P2001－508908A）
(43)公表日　平成13年７月３日(2001.7.3)

(51)Int.Cl.'	識別記号	Ｆ Ｉ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　9/30	３１０	Ｇ０６Ｆ　9/30	３１０Ｅ

審査請求　未請求　予備審査請求　未請求(全 49 頁)

(21)出願番号　特願平11－521337
(86)(22)出願日　平成10年９月21日(1998.9.21)
(85)翻訳文提出日　平成11年６月２日(1999.6.2)
(86)国際出願番号　ＰＣＴ／ＩＢ９８／０１４５３
(87)国際公開番号　ＷＯ９９／１８４８５
(87)国際公開日　平成11年４月15日(1999.4.15)
(31)優先権主張番号　９７２０３０３３．２
(32)優先日　平成９年10月２日(1997.10.2)
(33)優先権主張国　ヨーロッパ特許庁（ＥＰ）
(31)優先権主張番号　９７２０３９０５．１
(32)優先日　平成９年12月12日(1997.12.12)
(33)優先権主張国　ヨーロッパ特許庁（ＥＰ）

(71)出願人　コーニンクレッカ　フィリップス　エレクトロニクス　エヌ　ヴィ
オランダ国　5621　ベーアー　アインドーフェン　フルーネヴァウツウェッハ　1
(72)発明者　オーフステイン　アレクサンデル
オランダ国　5656　アーアー　アインドーフェン　プロフ　ホルストラーン　6
(72)発明者　デュクストラ　エールコ　イェー
オランダ国　5656　アーアー　アインドーフェン　プロフ　ホルストラーン　6
(74)代理人　弁理士　杉村　暁秀　（外２名）

最終頁に続く

(54)【発明の名称】　可変命令セットコンピュータ

(57)【要約】
ソースプログラムが、処理ユニット（100）のマイクロコントローラコア（114）上で実行される。該コア（114）は、マイクロコントローラ固有の命令の所定セットからのネイティブ命令を実行することができる。前処理ステップにおいて、上記ソースプログラムのプログラム命令文に対して、プログラム固有の仮想マシンが対応する仮想マシン命令のセットと共に次のように定義される。即ち、上記プログラム命令文の命令系列での表現が、ネイティブ命令のみを使用した場合に較べて少ない記憶空間しか必要としないように定義される。上記プログラム固有の仮想マシンに対して、該プログラム固有の仮想マシンの命令を上記コア（114）のネイティブ命令に変換するための関連する変換手段（132）が定義される。上記ソースプログラムの命令文は定義された仮想マシンの命令を含む命令の系列で表現される。この命令の系列は命令メモリ（120）に記憶される。変換手段（132）は処理ユニット（100）内で表される。実行の間に、命令メモリ（120）から命令が取り込まれる。そして、変換手段（132）は、上記の取り込まれた仮想マシン命

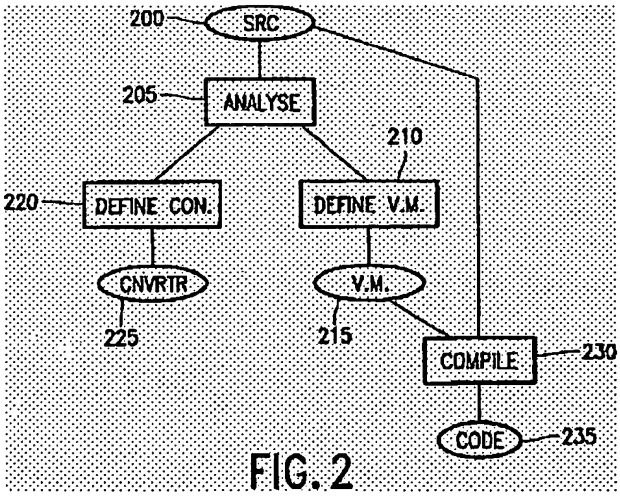

FIG. 2

【特許請求の範囲】

１．ソースプログラムを、所定のマイクロコントローラ固有の命令のセットからのネイティブ命令を実行する所定のマイクロコントローラコアを有するような処理ユニット上で実行する方法であって、該方法が、

－　前記ソースプログラムのプログラム命令文を仮想マシン命令を有する命令の系列で表現すると共に、該命令の系列を命令メモリに記憶する前処理ステップと、

－　前記命令メモリから命令を取り込むと共に、該命令メモリから取り込まれた仮想命令を前記処理ユニットの変換手段を用いてネイティブ命令に変換し、且つ、これらネイティブ命令を実行のために前記マイクロコントローラコアに供給するような実行ステップと、

を有するような方法において、前記前処理ステップが、

前記ソースプログラムのプログラム命令文に対して、対応する仮想マシン命令のセットを伴うプログラム固有の仮想マシンを、前記プログラム命令文の前記命令の系列での表現が該プログラム命令文を表現するのにネイティブ命令のみを使用した場合に較べて前記命令メモリ内に少ない記憶空間しか必要としないような形態で、定義し、

前記プログラム固有の仮想マシンに対して、該プログラム固有の仮想マシンの仮想マシン命令を前記マイクロコントローラコアのネイティブ命令に変換するような関連する変換手段を定義し、

前記関連する変換手段を前記処理ユニット内で表す、

ような各ステップを有していることを特徴とするソースプログラムを処理ユニット上で実行する方法。

２．請求項１に記載のソースプログラムを処理ユニット上で実行する方法において、前記前処理ステップが、

－　前記ソースプログラム内のプログラム命令文の複数の群を識別し、

－　これらプログラム命令文の群の各々に対して、

プログラム群固有の仮想マシンを対応する仮想マシン命令のセットと共

に定義し、

前記プログラム群固有の仮想マシンに対して、仮想マシン命令を前記マイクロコントローラコアのネイティブ命令に変換するような関連する変換手段を発生し、

前記関連する変換手段を前記処理ユニット内で表し、

前記処理ユニット内に、発生された各仮想マシン命令を前記群固有の変換手段に関連付ける選択データを記憶する、

ような各ステップを有し、前記実行ステップが、取り込まれた命令に対して前記選択データにより示される前記関連する変換手段を選択するステップを有していることを特徴とする方法。

3．請求項1に記載の方法において、該方法が、

他の仮想マシンの仮想マシン命令を含む命令を有するような命令モジュールを入力すると共に、前記処理ユニットの他の変換手段が前記他の仮想マシン命令をネイティブ命令に変換することができるようにする変換データを入力し、

前記命令モジュールを前記命令メモリに記憶し、

前記変換データを前記処理ユニットに記憶し、

前記他の仮想マシン命令の各々を前記変換データに関連付ける選択データを前記処理ユニットに記憶する、

ような各ステップを有し、前記実行ステップが、

取り込まれた他の仮想マシン命令に対して、前記選択データにより示される前記関連する変換データを選択し、

前記他の変換手段を前記選択された変換データの制御の下で動作させる、

ような各ステップを有していることを特徴とする方法。

4．仮想マシンの仮想マシン命令と呼ぶ命令を実行する処理ユニットであって、

所定のマイクロコントローラ固有の命令のセットからの、前記仮想マシン命令とは異なるネイティブ命令を実行する所定のマイクロコントローラコアと、

前記仮想マシン命令の少なくとも1つを含む命令を記憶する命令メモリと、

前記命令メモリから取り込まれた仮想マシン命令を、前記マイクロコントロ

ーラコアにより実行するための少なくとも１つのネイティブ命令に変換する変換手段を有するコンバータと、

を有するような処理ユニットにおいて、

前記コンバータが複数の異なる仮想マシンに対して前記変換を実行するように動作することを特徴とする処理ユニット。

５．請求項４に記載の処理ユニットにおいて、前記変換手段が再プログラム可能な形式のものであることを特徴とする処理ユニット。

６．請求項４に記載の処理ユニットにおいて、前記コンバータが前記複数の仮想マシンの各々に対して、対応する仮想マシンの仮想マシン命令を変換するための変換手段を有していることを特徴とする処理ユニット。

７．請求項６に記載の処理ユニットにおいて、該処理ユニットが前記命令メモリ内の少なくとも２つの分離された群のロケーションを各変換手段に関連付ける選択データを有し、前記コンバータが前記命令メモリ内のロケーションから取り込まれた命令を前記選択データに基づいて前記変換手段に選択的に指向させることを特徴とする処理ユニット。

８．請求項４に記載の処理ユニットにおいて、該処理ユニットは、前記ネイティブマシンと前記仮想マシンとの間及び／又は異なる仮想マシンの間の区別をするために、命令に対してメモリから関連する選択データを取り込むように動作し、前記コンバータが前記取り込まれた命令を前記選択データに基づいて変換手段に選択的に指向させる検出器を有していることを特徴とする処理ユニット。

９．請求項４に記載の処理ユニットにおいて、前記コンバータは１つの仮想マシン命令を１つの対応するネイティブ命令に変換するように動作し、前記仮想マシン命令は対応するネイティブ命令よりも前記命令メモリ内に要する記憶空間に関して一層コンパクトにコード化されていることを特徴とする処理ユニット。

10．請求項４に記載の処理ユニットにおいて、前記コンバータは１つの仮想マシン命令を複数のネイティブ命令の所定の系列に変換するように動作し、該処理

ユニットが、

前記コンバータと前記マイクロコントローラコアとの間に結合されて、前記ネイティブ命令の系列を前記マイクロコントローラコアに順次供給するシーケンサと、

上記供給の間に、前記命令メモリからの命令の取り込みを禁止する禁止手段と、

を有することを特徴とする処理ユニット。

11．請求項１０に記載の処理ユニットにおいて、前記禁止手段が前記禁止を前記マイクロコントローラコアの命令ポインタのインクリメントを妨害することにより実行するよう動作することを特徴とする処理ユニット。

12．請求項１１に記載の処理ユニットにおいて、該処理ユニットは、当該命令取り込み部の命令カウンタの変化に応答して前記命令メモリから命令を取り込む命令取り込み部を有し、前記命令カウンタは前記マイクロコントローラコアの命令ポインタの変化に応答して異なる値に設定され、前記禁止手段は前記命令カウンタの値の変化を妨害することにより前記禁止を実行するよう動作することを特徴とする処理ユニット。

【発明の詳細な説明】

可変命令セットコンピュータ

本発明は、マイクロコントローラ固有の命令の所定のセットからのネイティブ命令を実行するための所定のマイクロコントローラコアを有するような処理ユニット上でソースプログラムを実行する方法に係り、該方法が、

－　上記ソースプログラムのプログラム命令文を仮想マシン命令（仮想機械語命令）を有する命令の系列で表現すると共に該命令の系列を命令メモリに記憶する前処理ステップと、

－　上記命令メモリから命令を取り込むと共に、該命令メモリから取り込まれた仮想命令を前記処理ユニットの変換手段を用いてネイティブ命令に変換し、更に、これらネイティブ命令を実行のために前記マイクロコントローラコアに供給するような実行ステップと、

を有するような方法に関する。

本発明は、更に、仮想マシンの仮想マシン命令と呼ばれる命令を実行する処理ユニットであって、

マイクロコントローラ固有の命令の所定のセットからの上記仮想マシン命令とは異なるネイティブ命令を実行する所定のマイクロコントローラコアと、

上記仮想マシン命令の少なくとも１つを含む命令を記憶する命令メモリと、

上記命令メモリから取り込まれた仮想マシン命令を、上記マイクロコントローラコアにより実行するための少なくとも１つのネイティブ命令に変換するための変換手段を有するコンバータと、

を有するような処理ユニットにも関する。

益々、ソースプログラムは、該プログラムが実行されるべきマイクロコントローラコアのネイティブ命令の代わりに、仮想マシンの命令で表現される（コンパイルされる）ようになってきている。仮想マシンを用いる主たる理由は、異なるマシン（プラットフォーム）の間でのプログラムの移植性である。仮想マシンの仮想マシン命令で表現されたプログラムは、幾つかの具体的マシン上で動作する適切なインタプリタを用いることにより、これらマシン上で比較的容易に実行す

ることができる。現時点では、移植性のあるプログラムを使用しようとする推進力はジャバであり、ジャバプログラムはインターネットを介して交換されると共に、異なる命令セットを持つプロセッサを用いた異なるネイティブマシン上で実行することができる。コンパイラを用いて、ジャバプログラムは、ジャバ仮想マシンの命令を形成するジャババイトコード（ＪＢＣ）で表現される。結果としてのコードは、通常、ジャバアプレットと称される。

従来、仮想マシン命令で表されるプログラムはソフトウェア翻訳により実行されている。プロセッサ（ＣＰＵ）が特別なインタプリタプログラムを実行し、そこでは、ループ内で該プロセッサが仮想マシン命令を取り込み、該命令を当該プロセッサのマイクロコントローラコアのネイティブ命令の系列にデコードし、そして各ネイティブ命令を実行する。この技術は、遅く、しかも比較的大きな付加的なインタプリタプログラムを必要とする。実行速度を改善するため、所謂ジャスト・イン・タイム（ＪＳＴ）コンパイル技術が使用される。仮想マシン命令で表されたソフトウェアモジュールの実行を開始する直前に、該モジュールはネイティブコードにコンパイルされる（即ち、ネイティブマシン命令で表される）。このようにして、該モジュールは当該コンパイラ用のコードに加えて、２度記憶されねばならない。ソフトウェア翻訳の上記の付加的な記憶要件は、組み込み型のシステムにとっては望ましいものではない。性能及び記憶上のオーバーヘッドを避けるために、ハードウェアインタプリタを使用することが好ましい。ハードウェアインタプリタそれ自体は、ウォレンの抽象命令セット用のプロログ(Prolog)プリプロセッサ(pre-processor：前処理プログラム）の形で知られている。１９８６年のマイクロプロセッシング及びマイクロプログラミングの第７１～８１頁における、Ｂ・クネドラー及びW・ローゼンスティールによる“ウォレンの抽象命令セット用のプロログプリプロセッサ”なる論文には、プロログプログラミング言語で書かれたプログラムをモトローラの68000プロセッサ（MC68000）上で翻訳するためのプリプロセッサが記載されている。該プロログソースプログラムを命令に翻訳するためにコンパイラが使用され、これら命令はウォレン氏により定義されたもので、通常、プロログプログラムを実行するために使用され

る。これらのウォレン命令のセットは、プロログプログラムを実行するために設計された仮想マシンを形成する。上記コンパイルの結果としてのウォレン命令の系列はＲＡＭにロードされ、上記プリプロセッサの助けでMC68000により実行される。電源投入後、上記MC68000はネイティブなMC68000命令を実行することにより、先ずブート処理を行う。該ブート処理の後、該MC68000はプロログ命令の実行を開始することが可能となる。これは、所定のアドレス範囲にジャンプすることにより開始される。上記プリプロセッサはメモリにマップされる装置であり、該装置は上記範囲にマップされている。このプリプロセッサがアドレス指定されると、該プリプロセッサは自身のＲＡＭから（翻訳されたプロログプログラムの）ウォレン命令を読み出し、MC68000命令の系列及び定数を適応的に合成し、これらを実行のために直接ＣＰＵに送出する。各ウォレン命令に対するMC68000命令は上記プリプロセッサのＲＯＭに記憶されている。通常、該プリプロセッサは１つのウォレン命令を１つの系列のMC68000命令に翻訳する。該プリプロセッサは自身のＲＡＭコントローラ及びＲＯＭコントローラを含み、これらコントローラが当該プリプロセッサのＲＡＭ及びＲＯＭ用のアドレスを発生する。ＲＡＭコントローラはＲＡＭ命令のポインタを管理する。MC68000の順次の読み出し動作の各々の結果、プリプロセッサは上記系列の次の命令（及び随意の定数）をＣＰＵに送ることになる。当該系列が完了すると、次の読み取り動作の結果として、当該プログラムの次のウォレン命令に対応する系列の最初の命令がＣＰＵに送出される。この既知のプリプロセッサは１つの仮想マシン（ウォレンマシン）をサポートする。

本発明の一つの目的は、もっと柔軟性のある上述したような種類の方法及び処理ユニットを提供することにある。また、本発明の他の目的は、プログラムがもっとコンパクトな形で表される上述したような種類の方法及び処理ユニットを提供することにある。

本発明の上記目的を達成するため、本方法は、

前記ソースプログラムのプログラム命令文に対して、対応する仮想マシン命令のセットを伴うプログラム固有の仮想マシンを、上記プログラム命令文の前記命

令系列での表現が該プログラム命令文を表現するのにネイティブ命令のみを使用した場合に較べて前記命令メモリ内に少ない記憶空間しか必要としないような形態で定義し、

上記プログラム固有の仮想マシンに対して、該プログラム固有の仮想マシンの仮想マシン命令を前記マイクロコントローラコアのネイティブ命令に変換するような関連する変換手段を定義し、

該関連する変換手段を前記処理ユニット内で表す、

ような各ステップを有していることを特徴としている。

本発明によれば、プログラムに対してプログラム固有の仮想マシンが、該プログラムが前記コアのネイティブ命令で表現された場合よりも該プログラムが一層コンパクトな形で表現することができるように、定義される。また、関連する変換手段も定義される。この変換手段は、例えば、ＲＯＭに記憶された変換テーブル若しくはマイクロコード又はＰＬＤのような専用のロジックを使用して実施化することができる。定義された変換手段は、前記処理ユニット内で表される。このようにして、プログラムの高速実行が維持される一方、同時にコンパクトな表現が達成される。この方法は、組み込み型アプリケーションと共に使用するのに特に好適である。この場合、上記ソースプログラムは、当該組み込み型システム内で最初に表される全てのプログラム命令文に関係する（例えば、ユーザがシステムを購入した時に当該システム内に存在するプログラム）。該プログラムはＲＯＭのような永久メモリに記憶することもでき、又はEEPROMのような再プログラム可能なメモリに記憶することもできる。組み込み型アプリケーションの場合は、該組み込み型アプリケーションのプログラムを表すのに使用されるコードがコン

パクトであり、該コードを実行する性能が良好であることが高度に望まれる。典型的には、組み込み型アプリケーション用に用いられる処理ユニットはファミリー思想に基づくものであり、そこでは、特定のアプリケーションに対して、処理ユニットが、該アプリケーションに必要とされる所与のマイクロコントローラコア及びＩ／Ｏ又は記憶部品から作成される。経費削減のため、上記コア及び部品

に対しては何の又は少なくとも大きな変更はしないことが望ましい。本発明によれば、一方では、当該プログラムに対して特別に仮想マシン命令が定義されて、コードのコンパクト化を達成するための充分な柔軟性を与え、他方では、該プログラムを実行するために在庫品的な（即座に入手可能な）コアが使用される。仮想マシン命令からネイティブ命令への所要な変換は変換手段により実行され、該手段は一種のプリプロセッサと見ることができる。

１９９５年１月のＰＯＰＬ’９５の第３２２～３３２頁におけるトッド・Ａ・プロエブスティングによる“ＡＮＳＩ　Ｃインタプリタのスーパー演算子を用いた最適化”なる文献は、バイトコードによるインタプリタ用の最適化技術としてのスーパー演算子の使用について記載している。最初のステップとして、プログラムを仮想マシン命令で表すためにコンパイラが使用される。この仮想マシン命令は、該仮想マシン命令を用いたプログラムの表現が一層コンパクトになるように、選択される。このような仮想マシンの一例はスタックマシンであり、斯かるスタックマシンは、そのコンパクトな表現で知られている。次に、コンパイルされたコード内で頻繁に発生する仮想マシン命令の系列が識別され、新たに定義された追加の仮想マシン命令により置換されるが、その場合、例えば１つの新たな命令（所謂、スーパー演算子）が４つの既存の命令の系列を置換する。スーパー演算子は、コンパイラの構文解析相の間に、当該コンパイラの中間表現の間に構築される表現ツリーにおいて最も頻繁に発生する項目を識別することにより定義することもできる。限られた数（例えば、２５６を大きく下回る）のスーパー演算子のみが定義される。これにより、スーパー演算子のコンパクトなコード化（例えば、単一バイトのコードを用いる）が可能となる。この場合、該コードの範囲の

一部が元の仮想マシン命令を表すために割り当てられ、該範囲の一部は上記スーパー演算子を表す新たに追加した仮想マシン命令を表すために使用される。２０個のみのスーパー演算子を用いることにより、既に大幅なコードのコンパクト化が達成される。これら仮想マシン命令は、ＭＩＰＳ R3000又はＳＰＡＲＣプロセッサ上で走るソフトウェアインタプリタを用いて翻訳される。ハードウェアイン

タプリタは使用されず、これは所与の組み込み型マイクロコントローラコアとの組み合わせで、性能上の不利益なしに所与の組み込み型アプリケーションプログラムのコードのコンパクト化をもたらす。

請求項２に記載した実施例においては、異なる群のプログラム命令文に対して、良好なレベルのコードのコンパクト化を達成することができる。好ましくは、ネイティブ命令の同一の副セットに実質的に関係するプログラム命令文は一緒に群にまとめられ、斯かる群に対して仮想マシンが定義される。例えば、浮動小数点の演算に対するよりも、整数の演算に対しては異なる仮想マシンが定義される。ネイティブ命令の略同一の副セットに関係するが、これら命令をかなり相違する頻度で使用するプログラム命令文は群にまとめるのが好ましく、これにより最も頻繁に発生する命令（の系列）に対しては特別な仮想マシン命令の定義を可能にする。マルチタスクシステムの場合は、好ましくは、１つのタスクのプログラム命令文は１つの仮想マシンに関連付けられる。これは、組み込み型のプログラムの開発を簡単にする。何故なら、通常は１つのタスクに関するプログラム命令文は一人の技術者の管理の下で作成され、このやり方においては、上記管理の役割をコンパクト化されたコード及び関連する変換手段の制作の管理を含むように拡張することができるからである。選択された構成に依存して、実行されるべき仮想マシン命令に関連付けられた変換手段の選択が、実行の遅延を招き得る。この場合、個々の仮想マシン命令に対して変換手段を選択する代わりに、変換手段をタスクの切り換えの一部として切り換えることによりオーバーヘッドを低減することができる。この場合、同一のタスクを切れ目無く実行する限り、適切な変換手段を選択するためのオーバーヘッドが関わることはない。同一の仮想マシンは、

ネイティブ命令の同様の混合を同様の頻度で使用する幾つかのタスクに対しても使用することができることが分かるであろう。一方、タスク用のプログラムが２以上のプログラムモジュールを有し、その場合に、これらプログラムモジュールの少なくとも２つが、ネイティブ命令の異なる副セットに実質的に関係するか、又は同様の命令の副セットの命令を実質的に異なる頻度で使用するような場合は

、これらのモジュールに対しては異なる仮想マシンを使用するのが好ましい。同様に、オブジェクト指向のプログラミング技術が用いられる場合は、仮想マシンは同様のオブジェクトの各々又は群に対して定義することができる。

請求項３に記載の実施例においては、命令モジュールと、該モジュールの命令をネイティブ命令に変換するための変換データとが入力される。例えば、当該処理ユニットには追加の又は置換用のプログラムモジュールが供給される。該供給は、例えばプログラムモジュールを備えるＲＯＭを追加する又は交換することによりローカルなものであってもよく、又は該プログラムモジュールをフロッピーディスクのような可搬型記憶媒体からプログラム可能型メモリにロードすることでもよい。上記供給は、ローカルエリアネットワーク又はインターネットのような広域ネットワークの何れかのネットワークを介してなされてもよい。益々、組み込み型のシステムは、追加のソフトウェア、又は既存のソフトウェアの一部を当該システムの初期プログラミングの後で置換するようなソフトウェアを受け付けるように“開放型（open）”になってきている。上記初期ソフトウェアが専用の仮想マシンを用いて最適に圧縮されていた場合は、該マシンを新たに入力されたコードに対して使用することは、常に良好な結果を与えるとは限らず、不可能な場合さえある。元のソフトウェアの供給者が新たなソフトウェアを供給する場合でさえも、該ソフトウェアの性質が異なる場合があり、これは、該新たなソフトウェアに対しては別の仮想マシンを定義することにより一層良好なコンパクト化が達成される可能性があることを意味する。益々、入力されるソフトウェアが、元のソフトウェアとは完全に異なる性質のものであるようになってきている。一例として、新たなソフトウェアはジャバのバイトコードで表現されたジャバアプ

レットであるかもしれず、一方、元のソフトウェアは“Ｃ”で書かれており、元のプログラムに適合した仮想マシンと対応する変換手段とに最適にコンパイルされている。この定義された変換手段はジャババイトコードを変換するために使用することはできないことが理解されるであろう。何故なら、これらのコードは異なる仮想マシンを用いて表現されているからである。

本発明の前記目的を達成するため、本処理ユニットは、前記コンバータが複数の異なる仮想マシンに対して変換を実行するように動作することを特徴としている。

例えば、組み込み型のシステムには最初に或る仮想マシンを用いて表された組み込み型のプログラムが供給され、好ましくは、この仮想マシンは該特定のプログラムに対して定義される。後の段階で、当該ソフトウェアの略全部又は一部を取り替えるか又はソフトウェアモジュールを追加するようなソフトウェアの更新が必要となる。その場合に形成される全体のアプリケーションプログラムに対しては、その時点で有効なプログラム（依然として古いプログラムの部分も含むことができる）を一層良好に反映するような新たな仮想マシンを使用するのが望ましい。特に、ソフトウェアの所要の量は時間と共に増加する傾向にあるから、新たに生成されるアプリケーションプログラムに対しては一層高いレベルのコードのコンパクト化（異なる仮想マシンに反映されて）が必要であろう。このようにして、時間につれて異なる仮想マシンが使用され、その場合に、各時点では唯一の仮想マシンが使用される。他の例として、前述したように、異なる仮想マシンが組み込み型アプリケーションプログラムの異なる部分に対して同時に使用されるようにしてもよい。異なる仮想マシンは、“C”で書かれ且つプログラム固有の仮想マシンを用いてコンパクト化された組み込み型プログラム及び異なり且つ通常は予め決められている仮想マシンを用いて表現されたジャバアプレット等のプログラムのような、出所の異なるプログラムに対して使用することもできる。複数の仮想マシンは当該処理ユニット内に同時に存在することもでき、その場合には、時間につれて新たな仮想マシンが追加されるか又は既存の仮想マシンが置換されることが分かるであろう。

請求項５に記載の処理ユニットの実施例においては、前記変換手段は再プログラム可能な形式のものである。これによれば、新たなプログラムがロードされる場合に、新たな変換手段を当該処理ユニットに“ダウンロード”することができるようになる。上記の再プログラム可能な変換手段は、例えば、(E)EPROM等に記憶された再プログラム可能な変換テーブル若しくはマイクロコードを用いて、又

はＥ－ＰＬＤのような専用の再プログラム可能なロジックを用いて、実施化することができる。

請求項６に記載の処理ユニットの実施例においては、前記コンバータは仮想マシンの各々に対して専用の変換手段を有している。この実施例では、処理ユニットには幾つかの専用の変換手段が作成される。原理的に、１つの対応する変換手段を備える１つの大きな仮想マシンを使用することもできる。幾つかの仮想マシンを使用することにより、一層コンパクトなコードを達成することができる。例えば、組み込み型のプログラムが２つのプログラムモジュールにより形成されていると仮定して、これらのモジュールのうちの一方のモジュールは当該組み込み型システムのユーザインターフェースとしての面に主に関係し、かくして主にネイティブ整数命令を必要とし、第２モジュールは主に信号処理に関係して、主に浮動小数点命令を必要とする。更に、上記２つのモジュールの各々に対して最適の仮想マシンは２５６の仮想マシン命令を各々有し、これらモジュールが８ビットのコードを用いて表現されると仮定する。両モジュールをカバーする１つの仮想マシンは上記２つの仮想マシンの命令を合成することにより作成することができ、５１２までの（少なくとも２５６を越える）仮想マシン命令を得ることができる。結果として、該コードの大きさは増加する。何故なら、この場合は９ビットのコードが必要となるからである。このような増加は、通常、可能性のある命令の重なり合いにより、合成された変換テーブルは２つの別個の変換テーブルより小さいという事実によっては補償することはできない。

請求項７に記載の処理ユニットの実施例においては、前記コンバータは異なる

仮想マシンに対する変換手段の間を、実行されるべき仮想命令が記憶される命令メモリ内のロケーションに基づいて、区別する。例えば、ネイティブ命令とは別に２つの異なる仮想マシンの仮想マシン命令も使用される場合は、該メモリは３つの領域、即ち各形式の命令に対して１つの領域、に分割することができる。実行されるべき該命令のアドレスが何の領域に存在するかを判定することにより、上記コンバータは、変換が必要（ネイティブ命令が変換されるべき）か否か、及び何の変換手段が該変換に使用されるべきかを容易に検出することができる。

請求項８に記載の処理ユニットの実施例においては、ネイティブマシンと仮想マシンとの間及び／又は異なる仮想マシンの間を区別するために、別個の指示子（例えば、１以上のビット）が使用される。別個の指示子は各命令に対して使用することができる。他の例として、マシンの変更が生じる毎に設定されるような１つの包括的な指示子（例えば、レジスタ）を使用することもできる。

請求項９に記載の処理ユニットの実施例においては、１つの仮想マシン命令が一層コンパクトにコード化される１つの対応するネイティブ命令に変換される。このようにして、該変換は非常に単純となる。

請求項１０に記載の処理ユニットの実施例においては、１つの仮想マシン命令は複数のネイティブ命令の所定の系列に変換され、更なるレベルのコンパクト化を提供する。ネイティブ命令の該系列の前記マイクロコントローラコアへの供給を、例えば該系列が供給されている間は上記マイクロコントローラの命令ポインタ（プログラムカウンタ）のインクリメントを禁止すると共に、一連動作が完了した場合にインクリメントを可能にすることにより制御するために、シーケンサが使用される。他の例として、当該処理ユニットは、自身の命令カウンタの制御の下で前記命令メモリから命令を取り込む命令取り込み部を有する。一連動作が完了すると、該カウンタの変化が可能にされ、マイクロコントローラコアの命令ポインタ（プログラムカウンタ）の変化に応答して、該命令カウンタは異なる値に設定される。系列が処理されている間は、上記命令カウンタの値の変化は禁止される。

本発明の上記及び他の特徴は図を参照して説明される実施例から明らかになるであろう。

図１は、処理ユニット内にコンバータを配置する４つの可能性のある構造的選択例を示し、

図２は、仮想マシン及び関連するコンバータを規定する処理を示し、

図３は、プログラム命令文の幾つかのコヒーレントな群から形成されるプログラムに関する処理を示し、

図４は、コンバータのブロック図を示し、

図１は、処理ユニット１００内にコンバータを配置する４つの可能性のある構造的選択例を図示している。該処理ユニット１００の３つの主要部分は、マイクロコントローラ１１０、命令メモリ１２０及びプリプロセッサ１３０である。プリプロセッサ１３０はコンバータ１３２を有する。示される全ての図において、マイクロコントローラ１１０は命令メモリ１２０とプリプロセッサ１３０とを有している。処理ユニット１００それ自体は明示的には示されていない。マイクロコントローラ１１０内の全ての主たる要素を組み合わせることにより（好ましくはワンチップ装置とする）、最適な性能が達成される。もし所望なら、命令メモリ１２０及び／又はプリプロセッサ１３０はマイクロコントローラ１１０の外に配置することもでき、その場合は、マイクロコントローラバス１４０が該マイクロコントローラ１１０の外まで延長され、例えば、ＰＣＩのような外部バスに結合されるようにすることもできることが分かる。

命令メモリ１２０は、スタックマシン用の命令のような仮想マシン命令を含んでいる。該命令メモリはデータを記憶するためにも使用することができる。本発明は、データ及び命令が別れているハーバードアーキテクチャに限定されるものではない。マイクロコントローラ１１０は、マイクロコントローラに固有の命令の所定のセットからのネイティブ命令を実行するための、ネイティブマシンと呼ばれる、所定のマイクロコントローラコア１１４を備えるようなプロセッサ１１２を有している。組み込み型のソフトウェアを実行するのに適したマイクロコン

トローラの一例は、MIPS PR3001範囲のマイクロプロセッサのようなＲＩＳＣ型のマイクロコントローラである。上記プロセッサは、ネイティブ命令を、これら命令を実行する前に記憶するための命令キャッシュ１１６を有することができる。該マイクロコントローラコア１１４のネイティブ命令は、前記仮想マシンの仮想マシン命令とは異なる。マイクロコントローラ１１０それ自体は、命令メモリ１２０に記憶された仮想マシン命令を直接実行することはできない。命令を要求するプロセッサ１１２に応答して、プリプロセッサ１３０がネイティブ命令を出力する。該ネイティブ命令を発生することができるように、プリプロセッサ１３０は、取り込み手段１３４を用いて、命令メモリ１２０から仮想マシン命令を取

り込むことができる。プリプロセッサ１３０のコンバータ１３２は、命令メモリ１２０から取り込まれた仮想マシン命令を少なくとも１つのネイティブ命令に変換するために使用される。通常、仮想マシン命令はネイティブ命令の系列に変換される。プリプロセッサ１３０は、更に、上記系列のネイティブ命令を実行のためにマイクロコントローラコア１１４に供給する供給手段１３６を有している。仮想マシンプログラムを実行する場合、マイクロコントローラ１１０は、実際に、プリプロセッサ１３０により発生されたネイティブプログラムを実行する。マイクロコントローラ１１０の命令ポインタが通常命令メモリ１２０内の当該マイクロプロセッサ１１０により次に実行される必要のある次の命令を示す場合、ここでは、該命令ポインタはプリプロセッサ１３０に対して次のネイティブ命令（又は前の命令の再供給）が必要であることを示す。結果として、プリプロセッサ１３０は、命令メモリ１２０内の現在の（又は次の）仮想マシン命令を示す独立した仮想マシン命令ポインタを管理することになる。当該マイクロコントローラは、仮想マシン命令又は仮想マシン命令ポインタを（明確に）知る必要はない。

図１Ａにおいて、当該処理ユニットの主要な部分は、ＰＩバスのような汎用周辺装置相互接続バス１４０を介して相互接続されている。プリプロセッサ１３０は該バス上の周辺装置である。プリプロセッサ１３０はメモリにマップされた周辺装置として動作し、そこでは、該プリプロセッサに対し所定の範囲のアドレス

が割り当てられている。プロセッサ１１２がバス１４０上に上記範囲内のアドレスを伴う命令の要求を送出することに応答して、プリプロセッサ１３０はバス１４０上にネイティブ命令を出力する。必要があれば、プリプロセッサ１３０はバス１４０を介して命令メモリ１２０から仮想マシン命令を取り込む。

図１Ｂ及び図１Ｃにおいては、プリプロセッサ１３０はプロセッサ１１２と命令メモリ１２０との間に位置されている。プリプロセッサ１３０がネイティブ命令と仮想マシン命令とを区別する必要がある場合は、これらの構成は、命令メモリ１２０に記憶されたネイティブ命令の実行を遅延させることができる。明瞭化の理由で、図１Ａに示された全ての要素が図１Ｂ、図１Ｃ及び図１Ｄで繰り返されてはいない。

図１Ｄにおいては、プリプロセッサ１３０はプロセッサ１１２内に組み込まれている。プリプロセッサ１３０は、好ましくは、プロセッサ１１２の命令キャッシュ１１６とコア１１４との間に位置される。この構成は最適な性能を可能にするが、図１Ａ、１Ｂ及び１Ｃの構成とは違って、マイクロコントローラ１１０の変更を必要とし、プリプロセッサ１３０それ自体は、同一形式のコア１１４を備える異なる形式のプロセッサに対しては、在庫設計のようには、使用することはできない。

前処理ステップにおいては、当該処理ユニット１００により実行されるべきソースプログラムのプログラム命令文に対して、対応する仮想マシン命令のセットを伴うプログラム固有の仮想マシンが定義される。更に、上記のプログラム固有の仮想マシンに対して、対応するコンバータ１３２が仮想マシン命令をマイクロコントローラコアのネイティブ命令に変換するために定義される。該コンバータ１３２は、上記仮想マシン命令の各々に対して、ネイティブコア１１４の、該仮想マシン命令を実行するために必要とされる１個の又は通常は一連の対応するネイティブ命令を供給する。図２は上記仮想マシン及び関連するコンバータを定義する過程を図示している。ステップ２０５においては、ソースプログラム２００が解析される。この解析（例えば、演算の発生頻度）に基づいて、ステップ２１０ではプログラム固有の仮想マシン２１５が定義される。ステップ２２０においては、該仮想マシン用に、関連するコンバータ２２５が定義され、ステップ２３０においては上記ソースプログラムが該仮想マシンの命令で表され、結果としてコード２３５が得られる。上記仮想マシンは、前記プログラム命令文を表すのにネイティブ命令のみを使用するのに較べて、上記コードが命令メモリ内に必要とする記憶空間が少なくなるように、定義される。

好ましくは、上記のプログラム固有の仮想マシンは、上記プログラムを所定の仮想マシンの仮想マシン命令に変換することにより定義される。開始点として、コンパクトな表現が得られるものとして知られているスタックマシンのような仮想マシンが選択される。スタックマシンはオペランドを記憶するために明確なレジスタは使用しないが、その代わりに、オペランドはスタックに記憶され、演算

子は常に該スタックの最上要素（又は複数の要素）に作用するという事実により、スタックマシンの場合は、使用することが可能なオペランドの数はあまり限定されず、これらオペランドの管理が一層簡単である。これにより、スタックに基づくマシンはレジスタに基づくマシンの場合よりも、コンパイラを構築するのが容易となる。更に、スタックマシンの命令は、殆どのレジスタに基づくマシンよりも単純な構造を持つ傾向がある。しかしながら、好適なレジスタに基づくマシンを使用することもできることが分かるであろう。本発明を説明するため、付録Ａのスタックマシンが使用される。“Ｃ”プログラム言語で書かれたサンプルプログラム（８人の女王の問題を解く）が付録Ｂに示されている。該プログラムの付録Ａの仮想マシンのスタックコードへの変換が付録Ｃに示されている。該コードを解析することにより、頻繁に発生する命令の系列が識別され、各々がこれらの識別された系列の１つを表すような追加の仮想マシン命令が定義される。付録Ｄは、好適な追加のマシン命令のリストを示している。このような追加のマシン命令は、“スーパー仮想マシン命令”と見ることができる。次いで、上記プログラム固有の仮想マシンが、付録Ａの基本命令と付録Ｄのスーパー命令との組み合わせで形成される。付録Ｅは、付録Ｂのプログラムの付録Ａ及びＤの仮想マシンへ

の表現を示している。次に、上記のプログラム固有の仮想マシンに対して、図１に示すような関連するコンバータ１３２が、仮想マシン命令を前記マイクロコントローラコアのネイティブ命令に変換するために定義される。該コンバータ１３２は、上記仮想マシン命令（本例では、付録Ａ及びＤの命令の組み合わせ）の各々に対して、該仮想マシン命令をネイティブコア１１４上で実行するために要する１つの又は通常は一つの系列の対応するネイティブ命令を供給する。このような方法を用いれば、ネイティブ命令を表すのに通常１６又は３２ビットが使用されるのと較べて、各パラメータ化されていない命令に対して短い表現（例えば、７ビット）を可能にするような比較的少数の仮想マシン命令（本例では、３８個の基本命令及び４０個のスーパー命令）を用いることにより、上記コードのコンパクト化が達成される。上記スーパー命令を使用する結果、当該プログラムを表

すのに少ない命令しか必要とされなくもなる。本例では、当該プログラムを基本仮想マシンの命令で表すのには３５６の命令文が必要である。スーパー命令を使用すると、これが２６２の命令文だけが必要となるように低減される。これは、当該プログラムがネイティブ命令で表現され、そこではＭＩＰＳプロセッサに対しては６０８個の命令文が必要となるような状況に較べると有利である。

好ましくは、プログラムに固有な仮想マシンは以下のようにして定義される。ソースプログラムは、所定の仮想マシンに基づいて所謂表現ツリーの中間レベル表現に変換される。表現ツリーは、該仮想マシンの基本演算を表すノードにより構成される。該表現ツリーは、該ツリーのノードの基本演算を所定の順序（例えば、接尾辞順）で実行する結果としての計算を表す。当該プログラムの表現ツリーへの変換は、フレーザ及びハンソンの「ライト“C”コンパイラlcc」のような、コンパイラの構文解析相を用いて実行される。次に、該表現ツリーとコヒーレントなツリー断片（一連の演算を表す）とが識別される。これらツリー及び断片の各々は、スーパー命令により表されるべき候補である。前記基本仮想マシン命令から始めて、最もコードサイズの節約が得られるようなツリー及びツリー断片が、スーパー命令により表される。該スーパー命令は前記仮想マシンに追加される。当該プログラムは、節約を決定するために、該新たな仮想マシン命令のセットにより表される。新たなスーパー命令は、新たな命令を追加する余裕が依然としてある限り（例えば、仮想マシン命令がバイトとして記憶される必要がある場合は、命令の数は２５６に限定される）、又は最早節約が達成されなくなるまで、追加される。最後に、当該ソースプログラムは上記プログラム固有の仮想マシンの命令で表され、変換データが発生される。

当該プログラムが、一旦、仮想マシン命令で表されると、結果としてのコードは命令メモリ１２０に記憶される。組み込み型システムの場合は、該コードは通常ＲＯＭに記憶されるが、該ＲＯＭはマスクプログラムされたＲＯＭ、又はＰＲＯＭ若しくは(E)EPROMのようなプログラム可能なＲＯＭとする。同様にして、上記の発生された変換データは処理ユニット１００のコンバータ１３２内で表される。該コンバータ１３２は、例えば、ＲＯＭに記憶される変換テーブル又はマイ

クロコードを用いて実施化することができる。また、コンバータ１３２は、ＰＬＤのようなロジックを用いて実施化することもできる。再プログラム可能なコンバータの場合は、同様の方法を、(E)EPROM又はE-PLDのような各々の再プログラム可能な技術に基づいて使用することができる。

図３は本発明による他の実施例を示し、該図において、例えば“C”でプログラムされたソースプログラム３００はステップ３１０において、プログラムモジュール、オブジェクト又はタスク固有のコードのような幾つかのコヒーレントなプログラム命令文の群に分割される。図示のものは、モジュール３１２及び３１６である。明確な分割は必要ではなく、ソースプログラムが既に好適なモジュール構造で利用可能であるかもしれないことが分かるであろう。群３１２及び３１６の各々に対して、図２のステップ２０５及び２１０と同様の方法により、プログラム群固有の仮想マシンが、対応する仮想マシン命令のセットと共に定義される（ステップ３３０）。図示したものは、各仮想マシン３３２及び３３６である。ステップ３４０においては、各群３１２及び３１６のプログラム命令文が各仮想マシン３３２及び３３６の命令で表され、結果として各コードモジュール３４２

及び３４６が得られる。これらコードモジュールはステップ３５０において命令メモリ１２０に記憶される。ステップ３６０においては、上記プログラム群固有の仮想マシン３３２及び３３６に対して、仮想マシン命令をマイクロコントローラコアのネイティブ命令に変換するための変換手段３６２及び３６６が各々発生される。ステップ３７０においては、これら変換手段３６２及び３６６が当該処理ユニット１００内で、例えば該処理ユニット１００内に変換テーブル又は変換ロジックをプログラムすることにより表される。結果として、コンバータ１３２は上記プログラム群の各々に対して固有の変換手段を有することになる。処理ユニット１００が取り込まれた仮想命令に関連する群固有の変換手段を選択することができるように、ステップ３９０においては、選択データが当該処理ユニットに記憶される。実行の間には、仮想マシン命令が命令メモリ１２０から取り込まれる。この場合、上記選択データが当該仮想マシン命令が属する仮想マシンに関

連する変換手段を捜すために使用される。

図４はコンバータ１３２のブロック図を示し、該図においてコンバータ１３２は幾つかの変換手段（４００、４１０及び４２０が図示されている）を有している。これら変換手段の各々が、固有の仮想マシンの仮想マシン命令を変換するために使用される。該コンバータ１３２は、各々が完全な変換を実行することができるような完全に自律的な変換手段を有してもよいことが分かるであろう。他の例として、コンバータ１３２は、別個の変換データ（テーブルのような）の制御の下で仮想マシンの各々に対して何らかの共有ロジックを使用することもできる。当該処理ユニット１００は選択データ４３０を有する。該選択データは、命令メモリ１２０に記憶される、又は当該変換手段が（部分的に）データとして記憶される場合は該変換手段と組み合わせて記憶される等の如何なる好適な方法でも表すことができる。また、選択データ４３０はメモリからロードされるコンバータ１３２内のレジスタの形を採ることもできる。種々の形の選択データを使用することができる。

他の実施例においては、命令メモリ１２０のアドレス範囲が幾つかの副範囲に

分割される。一つのアドレス副範囲は１つの仮想マシンのみの仮想マシン命令を記憶するために確保される。好ましくは、一つのアドレス副範囲はネイティブ命令を記憶するためにも確保される。このようなネイティブ命令は、例えば、当該システムを初期化するか、又はドライバ若しくは組み込み型ソフトウェアアプリケーションの特別な部分のようなソフトウェアモジュールが最適な性能のためにネイティブ命令にコンパイルされるのを可能にするために使用される。この実施例では、前記選択データは、規定されたアドレス副範囲の各々に関して、命令取り込み部１３４により取り込まれた命令をネイティブ命令に変換するために上記変換手段４００、４１０及び４２０のうちの何れが使用されるべきかを示す。この目的のため、コンバータ１３２は、命令メモリ１２０内のロケーションから取り込まれた命令を上記選択データに基づいて変換手段４００、４１０又は４２０のうちの１つに選択的に指向させる検出器４４０を有している。ネイティブ命令も命令メモリ１２０内に記憶されている場合は、該検出器４４０は、これら命令

がマイクロコントローラコア１１４へ供給されるように供給器１３６に直接供給されるのを保証する。

上記判断が取り込まれた命令のアドレスに基づく代わりの例として、情報が該命令に直接関連付けられて記憶されるようにすることもできる。例えば、命令メモリ１２０への各エントリの１以上のビットが、仮想マシン間の及び／又はネイティブコードと仮想コードとの間の区別をつけるために確保されるようにすることもできる。例えば、命令が７ビットを要するような場合に２つの仮想マシンのみが使用されるとすると、８番目のビットは当該命令が属する仮想マシンと関連する変換手段とを示すために使用することができる。明らかに、このような技術はアドレスに基づく区別と組み合わせることもできる。

他の例として、上記選択データはレジスタに記憶されてもよく、その場合、該レジスタは仮想マシンの間の切り換えが発生する毎に、別の変換手段を示すように設定される。該レジスタを設定するためには、特別な命令（例えば、ジャンプ命令の一形態）を使用することができる。斯かる構成をサポートする命令は、命令メモリ内の仮想マシン命令／ネイティブ命令と混ぜることができる。

上述したように、典型的には、仮想マシン命令はネイティブ命令の系列に変換される。命令の前記コアへの供給を調節するために、当該処理ユニット１００は、コンバータ１３２とマイクロコントローラコア１１４との間に結合されて上記ネイティブ命令の系列をマイクロコントローラコア１１４に順次供給するシーケンサ４５０を有する。該シーケンサ４５０はカウンタのような通常の部品を用いて実施化することが可能であり、該カウンタは、例えば、当該マイクロコントローラコアからの新たな命令が必要であることを示すトリガ（例えば、当該コアの命令カウンタのインクリメント）の結果としてインクリメントされる。通常の処理ユニットにおいては、マイクロコントローラコアの命令ポインタ（プログラムカウンタとも称される）の変化の結果として、取り込み部１３４は命令メモリ１２０から命令を取り込み、供給器１３６は該命令をマイクロコントローラコア１１４に供給する。自動的な取り込みと供給との間の結合を絶つため、当該処理ユニット１００は、更に、前記系列の供給の間は命令メモリからの命令の取り込み

を禁止するための禁止手段４６０を有している。

上記順次供給及び禁止は幾つかの方法で実施することができるこあろう。他の実施例においては、禁止手段４６０は上記禁止をマイーラコア１１４の命令ポインタのインクリメントを妨害することによう動作する。これには、コア１１４に対する小さな変更を必要と、コア１１４への追加の制御ラインにより命令カウンタの選択的祭り、その場合、上記禁止手段４６０は前記系列からの命令が供給さはインクリメントを禁止し、系列が完全に供給されたらインクリメする。

他の実施例においては、命令取り込み部１３４が、マイクロコン１１４の命令ポインタとは別に、自身の命令カウンタを維持する。０は、何時マイクロコントローラの命令ポインタのインクリメント的には変化）の結果、命令取り込み部１３４の命令カウンタ１３５を制御する。上述したのと同様に、禁止手段４６０は前記系列から

されている場合は命令カウンタ１３５の変化を禁止し、系列が完全ら変化を可能にする。命令カウンタ１３５の変化の結果、命令取りは、通常、命令メモリ１１２の該命令カウンタにより示されるアドな命令を取り込む。有利には、仮想マシン命令のコンパクトな表現、命令取り込み部１３４はメモリから数個の命令を１つの読み出しむことができる（例えば、４つの１バイト命令を１つの３２ビット読み出すことができる）。このように、命令カウンタ１３５の全てしも結果として新たな命令の実際の取り込みになる必要はない。

本発明による他の実施例においては、他の仮想マシンの仮想マシれたプログラムモジュールが、例えば、ネットワークを介して又はクグラウンドメモリから入力される。このようなプログラムモジュ、ジャバアプレットである。入力されたプログラムモジュールは命０に記憶される。該他の仮想マシン命令のネイティブ命令への変換めるため、変換データも入力される。該変換データは、例えば、テ

く変換手段用の変換テーブル、又はＰＬＤに基づく変換手段用のＥ－ＰＬＤプログラミングデータを規定する。該変換データは、コンバータ１３２の他の変換手段による後の使用のために、当該処理ユニットに記憶される。実行の間に適切な変換手段を選択することができるように、他の各仮想命令を上記変換データに関連付ける選択データも当該処理ユニットに記憶される。実行の間には、取り込まれた他の仮想マシン命令に関して、上記他の変換手段は上記選択データにより示される変換データの制御の下で動作される。

上述したように、コンバータ１３２は仮想マシン命令をネイティブ命令の系列に変換するためのテーブルを有することができる。一次元テーブルを使用することができ、その場合、該テーブルの各セルは、対応する１つの仮想マシン命令に対して一連のネイティブ命令を有する。セル番号が、対応する仮想マシン命令の値に対応してもよい。一例として、ジャバの整数加算（０ｘ６０）用のネイティブ命令の系列は、セル９６（１６進で０ｘ６０に等しい）内に配置することができる。ネイティブ命令の系列の長さは種々の仮想命令に対して大幅に変化するので、好ましくは、斯かる系列は明確なセル無しに一次元テーブル内に配置され、その場合、各系列が互いに直後に続くようにする。このような変換テーブル５００が図５に示されており、該図では暗黙のセル境界が破線を用いて示されている。ある仮想マシン命令に対する系列を捜すことができるように、コード索引テーブル５１０を使用することができ、該索引テーブルは各仮想マシン命令（ＶＭＩ１ないしＶＭＩＮ）に対して、当該変換テーブル５００内の対応する系列の開始点を示す。変換テーブル５００におけるＶＭＩ３に対応するセルに関して、ネイティブ命令ＮＩ１ないしＮＩＭからなる関連する系列５２０が図示されている。

変換に関する他の例がジャバのバイトコードbipush n（バイトｎを符号拡張し、結果をスタックの最上部に配置するために使用される）に関して示される。この仮想マシン命令は２つのバイト {0x16及びn} からなり、ここで、最初のバイトは演算を定義し、第２バイトはパラメータｎを提供する。該命令はネイティブＭＩＰＳ命令の下記の系列に変換することができる。

```
        ori $a0, $0, n/* Load register $a0 with constant n */
        sll $a0, $a0, 24/* Shift left by 24 bits */
        sra $a0, $a0, 24/* Arithmetic shift right, causing sign extension,
by
        /* replicating last left-most bit */
        sw $a0, 0 ($tosp)/* Store result at new top of stack */
        addi $tosp, -4/* Increment stack size */
```

この例は、仮想マシン命令がパラメータ化され、そこでは、演算コードには少なくとも１つのオペランドが続くことを示している。有利には、コンバータ１３２は変換テーブル５００を有し、そこでは、ネイティブ命令は完全なコードにより、又は命令スケルトンにより、のいずれかで表される。一例として、上記命令addi $tosp，-4（上記例の系列の最後の命令）は変数部分を有さず、したがって当該テーブル内に４バイトエントリとして全て配置することができる。上記命令ori

$a0，$0，n（上記例の系列の最初の命令）は変数部分を含んでおり、したがって該変数部分（ｎである）を特定せずに当該テーブル内にスケルトンとして配置される。好ましくは、命令スケルトン用の当該テーブルへのエントリは完全命令（例えば、ＭＩＰＳプロセッサの場合は４バイト）と同じ幅とし、一様なテーブルを可能とする。該テーブル（又は他の（複数の）テーブル）には、どの様にしてネイティブ命令スケルトンの特定されていない部分を埋めるべきかを示す他の情報を配置することができる。有利には、上記の特定されていない部分を埋めるために、マイクロプログラミングが使用される。この場合、上記の他の情報はマイクロコードを有するか又はマイクロコードを示す。命令スケルトンに対しては完全ネイティブ命令と同じ構成（幅及び組立）を使用するのが有利であることが分かるであろう。しかしながら、他の構成も同様に使用することができる。

当該仮想マシンがスタック指向のマシンである場合は、好ましくは、スタック又は該スタックの少なくとも最上側の要素はマイクロコントローラ１１０のレジスタ上にマップされるようにする。このようにして、メモリスタック（仮想マシ

ンスタックを伴う）はレジスタスタックにマップされる。レジスタ$r1、$r2及び$r3が当該メモリスタックの３つの連続した要素を含んでおり、その場合に、最初は$r1が該メモリスタックの第１の空のロケーション（該スタックの最上部の上）に対応し、$r2が該メモリスタックの最上部を含み、$r3が該メモリスタックの第２の要素を含んでいると仮定すると、ジャバのバイトコードbipush nはネイティブＭＩＰＳ命令の下記系列に変換することができる。

```
ori $r1, $0, n
sll $r1, $r1, 24
sra $r1, $r1, 24
```

この演算の後、$r1は当該メモリスタックの最上部を含むようになる。

同様にして、整数加算（0x60）用のジャババイトコード（仮想マシン命令）は、最初は$r1が該メモリスタックの第１の空のロケーション（該スタックの最上部の上）に対応し、$r2が該メモリスタックの最上部を含み、$r3が該メモリスタッ

クの第２の要素を含んでいるような同一の位置から始めて、ＭＩＰＳ命令の下記系列に変換することができる。

```
add $r3, $r2, $r3
```

この演算の後、$r3は当該メモリスタックの最上部を含むようになる。

上記例において、好ましくは、メモリスタックの最上部の位置（即ち、どのレジスタが当該メモリスタックの最上部を含むか）は、コンバータ１３２のレジスタ１３８を用いて示されるようにする。該コンバータは、レジスタスタックポインタ（ＲＳＰ）と呼ばれるレジスタ１３８を、適切なネイティブ命令を発生するために使用する。好ましくは、ネイティブ命令のレジスタオペランドを特定するためにマイクロプログラミングが使用される。このようにして、固定のネイティブ命令も可変となる。何故なら、レジスタオペランドはコンバータ１３２により特定される必要があるからである。好ましくは、このようなオペランドも命令スケルトンを用いて変換テーブル５００に記憶されるようにする。ＲＳＰが最初の

空きのレジスタを指すと仮定すると、ジャババイトコードhipush nは、対応するマイクロコードの制御の下でネイティブＭＩＰＳ命令の下記の系列に変換することができる。

マイクロコード命令

```
rsp = 1;                    f_tg = rsp + lori $(rsp + 1), $0, n
f_tg = rsp + 1; f_a0 = rsp + lsll $(rsp + 1), $(rsp + 1), 24
f_tg = rsp + 1; f_a0 = rsp + lsra $(rsp + 1), $(rsp + 1), 2
```

ここで、f_{tg}は当該命令用の目標レジスタを示し、f_{a0}及びf_{a1}は当該命令用の第１及び第２引数レジスタを各々示す。スタックの最上側の２つの要素を加算するための、続くジャババイトコードiaddは、結果として、下記のマイクロコード及び命令となる。

```
f_tg = rsp + 2; f_a0 = rsp + 2; f_a1 = rsp + 1; rsp += ladd $(rsp + 2), $(rsp +
2), $(rsp + 1)
```

付録Ａ、基本仮想マシン命令

演算子	形式接尾辞	演算
ＡＤＤＲＦ	Ｐ	パラメータのアドレス
ＡＤＤＲＧ	Ｐ	グローバルのアドレス
ＡＤＤＲＬ	Ｆ	ローカルのアドレス
ＣＮＳＴ	ＣＳＩＵＰＦＤ	定数
ＢＣＯＭ	Ｕ	ビット単位補数
ＣＶＣ	ＩＵ	文字からの変換
ＣＶＤ	ＩＦ	倍からの変換
ＣＶＦ	Ｄ	浮動からの変換
ＣＶＩ	ＣＳＵＤ	整数からの変換
ＣＶＰ	Ｕ	ポインタからの変換
ＣＶＳ	ＩＵ	短からの変換
ＣＶＵ	ＣＳＩＰ	無符号からの変換
ＩＮＤＩＲ	ＣＳＩＰＦＤＢ	取り込み

NEG	IFD	否定
ADD	IUPFD	加算
BAND	U	ビット単位のアンド
BOR	U	ビット単位のオア
BXOR	LT	ビット単位の排他的オア
DIV	IUFD	除算
LSH	IU	左シフト
MOD	IU	モジュロ
MUL	IUFD	乗算
RSH	IU	右シフト
SUB	IUPFD	減算
ASGN	CSIPFDB	割り当て
EQ	IFD	等しいならジャンプ
GE	IUFD	大きい又は等しいならジャンプ
GT	IUFD	大きいならジャンプ
LE	IUFD	小さい又は等しいならジャンプ
LT	IUFD	小さいならジャンプ
NE	IUFD	等しくないならジャンプ
ARG	IPFDB	引き数
CALL	IFDBV	関数呼び出し
RET	IFD	関数からの戻り
JUMP	V	無条件ジャンプ
LABEL	V	定義
ADDSP		ｽﾀｯｸﾎﾟｲﾝﾀのｲﾝｸﾘﾒﾝﾄ
POP		スタックをポップ
接尾辞		
B	ブール式	
C	文字	

D	倍
I	整数
F	浮動小数点
P	ポインタ
S	短
U	無符号整数
V	ラベル

付録Ｂ．サンプル“Ｃ”プログラム

```
int up[15], down[15], rows[8], x[8];
int queens(), print();

main()
{
        int i;
        {
                char *i;
                printf ("%s", i);
        }
        for (i = 0; i < 15; i++)
                up[i] = down[i] = 1;
        for (i = 0; i < 8; i++)
                rows[i] = 1
        queens(0);
        return 0;
}

   queens(c)
{
        int r;

        for (r = 0; r < 8; r++)
                if (rows[r] && up[r-c+7] && down[r+c]) {
                        rows[r] = up[r-c+7] = down[r+c] = 0;
                        x[c] = r;
                        if (c == 7)
                                print();
                        else
                                queens(c + 1);
                        rows[r] = up[r-c+7] = down[r+c] = 1;
                }
```

```
}

print()
{
        int k;

        for (k = 0; k < 8; k++) printf ("%c ", x [k] + '1');
        printf("\n");
}
```

付録Ｃ、デフォルトの仮想マシンで表されたサンプルプログラム

```
1   .text
2   addrgp(_main)
3   jumpv
4   .globl _main
5   _main:
6   addSP (-12)
7   addrlp(-4)
8   cnsti(0)
9   asgni
10  L.2:
11  addrlp(-8)
12  addrlp(-4)
13  indiri
14  cnsti(2)
15  lshi
16  asgni
17  addrlp(-12)
18  cnsti(1)
19  asgni
20  addrlp(-8)
21  indiri
22  addrgp(_down)
23  addp
24  addrlp(-12)
25  indiri
26  asgni
27  addrlp(-8)
28  indiri
29  addrgp(_up)
30  addp
31  addrlp(-12)
32  indiri
33  asgni
34  L.3:
35  addrlp(-4)
36  addrlp(-4)
37  indiri
38  cnsti(1)
39  addi
40  asgni
41  addrlp(-4)
42  indiri
43  cnsti(13)
44  lti(L.2)
45  addrlp(-4)
46  cnsti(0)
47  asgni
48  L.6:
49  addrlp(-4)
50  indiri
51  cnsti(2)
52  lshi
53  addrgp(_rows)
54  addp
55  cnsti(1)
56  asgni
57  L.7:
58  addrlp(-4)
59  addrlp(-4)
60  indiri
61  cnsti(1)
62  addi
63  asgni
64  addrlp(-4)
65  indiri
66  cnsti(8)
67  lti(L.6)
68  cnsti(0)
69  argi
70  addrgp(_queens)
71  calli
72  addSP(4)
73  pop
74  cnsti(0)
75  ret
76  L.1:
77  cnsti(99)
78  ret
79  .globl _queens
80  _queens:
81  addSP (-52)
82  addrlp(-4)
83  cnsti(0)
84  asgni
85  L.13:
86  addrlp(-8)
87  addrlp(-4)
```

88	indiri	119	addrgp(_up+28)	150	addrlp(-36)
89	asgni	120	addp	151	cnsti(0)
90	addrlp(-12)	121	indiri	152	asgni
91	cnsti(2)	122	addrlp(-16)	153	addrlp(-24)
92	asgni	123	indiri	154	indiri
93	addrlp(-16)	124	eqi(L.17)	155	addrlp(-28)
94	cnsti(0)	125	addrlp(-8)	156	indiri
95	asgni	126	indiri	157	addi
96	addrlp(-8)	127	addrlp(-20)	158	addrlp(-32)
97	indiri	128	indiri	159	indiri
98	addrlp(-12)	129	addi	160	lshi
99	indiri	130	addrlp(-12)	161	addrgp(_down)
100	lshi	131	indiri	162	addp
101	addrgp(_rows)	132	lshi	163	addrlp(-36)
102	addp	133	addrgp(_down)	164	indiri
103	indiri	134	addp	165	asgni
104	addrlp(-16)	135	indiri	166	addrlp(-24)
105	indiri	136	addrlp(-16)	167	indiri
106	eqi(L.17)	137	indiri	168	addrlp(-28)
107	addrlp(-20)	138	eqi(L.17)	169	indiri
108	addrfp(8)	139	addrlp(-24)	170	subi
109	indiri	140	addrlp(-4)	171	addrlp(-32)
110	asgni	141	indiri	172	indiri
111	addrlp(-8)	142	asgni	173	lshi
112	indiri	143	addrlp(-28)	174	addrgp(_up+28)
113	addrlp(-20)	144	addrfp(8)	175	addp
114	indiri	145	indiri	176	addrlp(-36)
115	subi	146	asgni	177	indiri
116	addrlp(-12)	147	addrlp(-32)	178	asgni
117	indiri	148	cnsti(2)	179	addrlp(-24)
118	lshi	149	asgni	180	indiri

```
181  addrlp(-32)
182  indiri
183  lshi
184  addrgp(_rows)
185  addp
186  addrlp(-36)
187  indiri
188  asgni
189  addrfp(8)
190  indiri
191  cnsti(2)
192  lshi
193  addrgp(_x)
194  addp
195  addrlp(-4)
196  indiri
197  asgni
198  addrfp(8)
199  indiri
200  cnsti(7)
201  nei(L.21)
202  addrgp(_print)
203  calli
204  addSP(0)
205  pop
206  addrgp(L.22)
207  jumpv
208  L.21:
209  addrfp(8)
210  indiri
211  cnsti(1)
212  addi
213  argi
214  addrgp(_queens)
215  calli
216  addSP(4)
217  pop
218  L.22:
219  addrlp(-40)
220  addrlp(-4)
221  indiri
222  asgni
223  addrlp(-44)
224  addrfp(8)
225  indiri
226  asgni
227  addrlp(-48)
228  cnsti(2)
229  asgni
230  addrlp(-52)
231  cnsti(1)
232  asgni
233  addrlp(-40)
234  indiri
235  addrlp(-44)
236  indiri
237  addi
238  addrlp(-48)
239  indiri
240  lshi
241  addrgp(_down)
242  addp
243  addrlp(-52)
244  indiri
245  asgni
246  addrlp(-40)
247  indiri
248  addrlp(-44)
249  indiri
250  subi
251  addrlp(-48)
252  indiri
253  lshi
254  addrgp(_up+28)
255  addp
256  addrlp(-52)
257  indiri
258  asgni
259  addrlp(-40)
260  indiri
261  addrlp(-48)
262  indiri
263  lshi
264  addrgp(_rows)
265  addp
266  addrlp(-52)
267  indiri
268  asgni
269  L.17:
270  L.14:
271  addrlp(-4)
272  addrlp(-4)
273  indiri
```

```
274   cnsti(1)
275   addi
276   asgni
277   addrlp(-4)
278   indiri
279   cnsti(8)
280   lti(L.13)
281   L.12:
282   cnsti(99)
283   ret
284   .globl _print
285   _print:
286   addSP (-4)
287   addrlp(-4)
288   cnsti(0)
289   asgni
290   L.37:
291   addrlp(-4)
292   indiri
293   cnsti(2)
294   lshi
295   addrgp(_x)
296   addp
297   indiri
298   cnsti(49)
299   addi
300   argi
301   addrgp(L.41)
302   argp
303   addrgp(_printf)
304   calli
305   addSP(8)
306   pop
307   L.38:
308   addrlp(-4)
309   addrlp(-4)
310   indiri
311   cnsti(1)
312   addi
313   asgni
314   addrlp(-4)
315   indiri
316   cnsti(8)
317   lti(L.37)
318   addrgp(L.42)
319   argp
320   addrgp(_printf)
321   calli
322   addSP(4)
323   pop
324   L.36:
325   cnsti(99)
326   ret
327   .textEnd
328   .data
329   .globl _x
330   .align 4
331   _x:
332   .space 32
333   .globl _rows
334   .align 4
335   _rows:
336   .space 32
337   .globl _down
338   .align 4
339   _down:
340   .space 60
341   .globl _up
342   .align 4
343   _up:
344   .space 60
345   .dataEnd
346   .data
347   .align 1
348   L.42:
349   .byte 10
350   .byte 0
351   .align 1
352   L.41:
353   .byte 37
354   .byte 99
355   .byte 32
356   .byte 0
357   .dataEnd
```

付録Ｄ．スーパー命令

```
1     instr112 :  ADDRLP [-4]
2     instr123I :  INDIRI(ADDRLP [#])
3     instr298 :  EQI [L.17] (INDIRI (ADDP(*, *)), INDIRI(ADDRLP[-16]))
4     instr113 :  CNSTI [0]
5     instr133 :  CNSTI [1]
6     instr315 :  INDIRI (ADDRFP [8])
7     instr124 :  CNSTI [2]
8     instr249U :  ASGNI (ADDP(LSHI(*, *), ADDRGP[#]), *)
9     instr121 :  ADDRLP [-8]
10    instr132 :  ADDRLP [-12]
11    instr259U :  LTI [#](INDIRI(ADDRLP [-4]) ,CNSTI[8])
12    instr180 :  ASGNI(ADDRLP[-4]ADDI(INDIRI(ADDRLP [-4]), *))
13    instr160 :  ASGNI(ADDP(*, ADDRGP[_down]), INDIRI(*))
14    instr355 :  ADDRLP [-36]
15    instr471 :  ADDRLP [-52]
16    instr243 :  ASGNI(ADDP(LSHI(INDIRI(*), *), ADDRGP[_rows]), *)
17    instr352 :  ADDRLP [-24]
18    instr468 :  ADDRLP [-40]
19    instr354 :  ADDRLP [-32]
20    instr470 :  ADDRLP [-48]
21    instr514 :  ADDI (INDIRI(ADDP(*, ADDRGP[_x])), CNSTI[49])
22    instr211 :  LTI [L.2](INDIRI(ADDRLP[-4]), CNSTI[15])
23    instr313 :  ADDRLP [-20]
24    instr353 :  ADDRLP [-28]
25    instr461 :  NEI [L.21](INDIRI(ADDRFP[8]), CNSTI[7])
26    instr469 :  ADDRLP [-44]
27    instr114 :  ASGNI (ADDRLP[-4], CNSTI [0])
28    instr267 :  ADDRGP [_queens]
29    instr509 :  ADDRGP [_printf]
30    instr122 :  INDIRI ( ADDRLP [-4] )
31    instr336 :  ADDRGP [_up+28]
32    instr146 :  INDIRI ( ADDRLP [-8] )
```

```
33      instr147 :  ADDRGP [_down]
34      instr206 :  ADDRGP [_up]
35      instr219 :  ADDRGP [_rows]
36      instr276 :  ADDRLP [-16]
37      instr473 :  ADDRGP [_print]
38      instr474 :  ADDRGP [L.22]
39      instr515 :  ADDRGP [L.41]
40      instr523 :  ADDRGP [L.42]
```

付録Ｅ．プログラム固有の仮想マシンで表されるサンプルプログラム

```
1    .text
2    addrgp(_main)
3    jumpv
4    .globl _main
5    _main:
6    addSP (-12)
7    instr114
8    L.2:
9    instr121
10   instr122
11   instr124
12   lshi
13   asgni
14   instr132
15   instr133
16   asgni
17   instr146
18   instr132
19   instr160
20   instr146
21   instr206
22   addp
23   instr132
24   indiri
25   asgni
26   L.3:
27   instr133
28   instr180
29   instr211
30   instr114
31   L.6:
32   instr112
33   instr124
34   instr133
35   instr243
36   L.7:
37   instr133
38   instr180
39   instr259U(L.6)
40   instr113
41   argi
42   instr267
43   calli
44   addSP(4)
45   pop
46   instr113
47   ret
48   L.1:
49   cnsti(99)
50   ret
51   .globl _queens
52   _queens:
53   addSP (-52)
54   instr114
55   L.13:
56   instr121
57   instr122
58   asgni
59   instr132
60   instr124
61   asgni
62   instr276
63   instr113
64   asgni
65   instr146
66   instr132
67   indiri
68   lshi
69   instr219
70   instr298
71   instr313
72   instr315
73   asgni
74   instr146
75   instr313
76   indiri
77   subi
78   instr132
79   indiri
80   lshi
81   instr336
82   instr298
83   instr146
84   instr313
85   indiri
86   addi
87   instr132
88   indiri
89   lshi
90   instr147
```

91	instr298	123	addp	155	instr122
92	instr352	124	instr355	156	asgni
93	instr122	125	indiri	157	instr469
94	asgni	126	asgni	158	instr315
95	instr353	127	instr352	159	asgni
96	instr315	128	instr354	160	instr470
97	asgni	129	indiri	161	instr124
98	instr354	130	instr355	162	asgni
99	instr124	131	indiri	163	instr471
100	asgni	132	instr243	164	instr133
101	instr355	133	instr315	165	asgni
102	instr113	134	instr124	166	instr468
103	asgni	135	instr122	167	indiri
104	instr352	136	instr249U(_x)	168	instr469
105	indiri	137	instr461	169	indiri
106	instr353	138	instr473	170	addi
107	indiri	139	calli	171	instr470
108	addi	140	addSP(0)	172	indiri
109	instr354	141	pop	173	lshi
110	indiri	142	instr474	174	instr471
111	lshi	143	jumpv	175	instr160
112	instr355	144	L.21:	176	instr468
113	instr160	145	instr315	177	indiri
114	instr352	146	instr133	178	instr469
115	indiri	147	addi	179	indiri
116	instr353	148	argi	180	subi
117	indiri	149	instr267	181	instr470
118	subi	150	calli	182	indiri
119	instr354	151	addSP(4)	183	lshi
120	indiri	152	pop	184	instr336
121	lshi	153	L.22:	185	addp
122	instr336	154	instr468	186	instr471

```
187  indiri
188  asgni
189  instr468
190  instr470
191  indiri
192  instr471
193  indiri
194  instr243
195  L.17:
196  L.14:
197  instr133
198  instr180
199  instr259U(L.13)
200  L.12:
201  cnsti(99)
202  ret
203  .globl _print
204  _print:
205  addSP (-4)
206  instr114
207  L.37:
208  instr122
209  instr124
210  lshi
211  instr514
212  argi
213  instr515
214  argp
215  instr509
216  calli
217  addSP(8)
218  pop
219  L.38:
220  instr133
221  instr180
222  instr259U(L.37)
223  addrgp(L.42)
224  argp
225  instr509
226  calli
227  addSP(4)
228  pop
229  L.36:
230  cnsti(99)
231  ret
232  .textEnd
233  .data
234  .globl _x
235  .align 4
236  _x:
237  .space 32
238  .globl _rows
239  .align 4
240  _rows:
241  .space 32
242  .globl _down
243  .align 4
244  _down:
245  .space 60
246  .globl _up
247  .align 4
248  _up:
249  .space 60
250  .dataEnd
251  .data
252  .align 1
253  L.42:
254  .byte 10
255  .byte 0
256  .align 1
257  L.41:
258  .byte 37
259  .byte 99
260  .byte 32
261  .byte 0
262  .dataEnd
```

【図１】

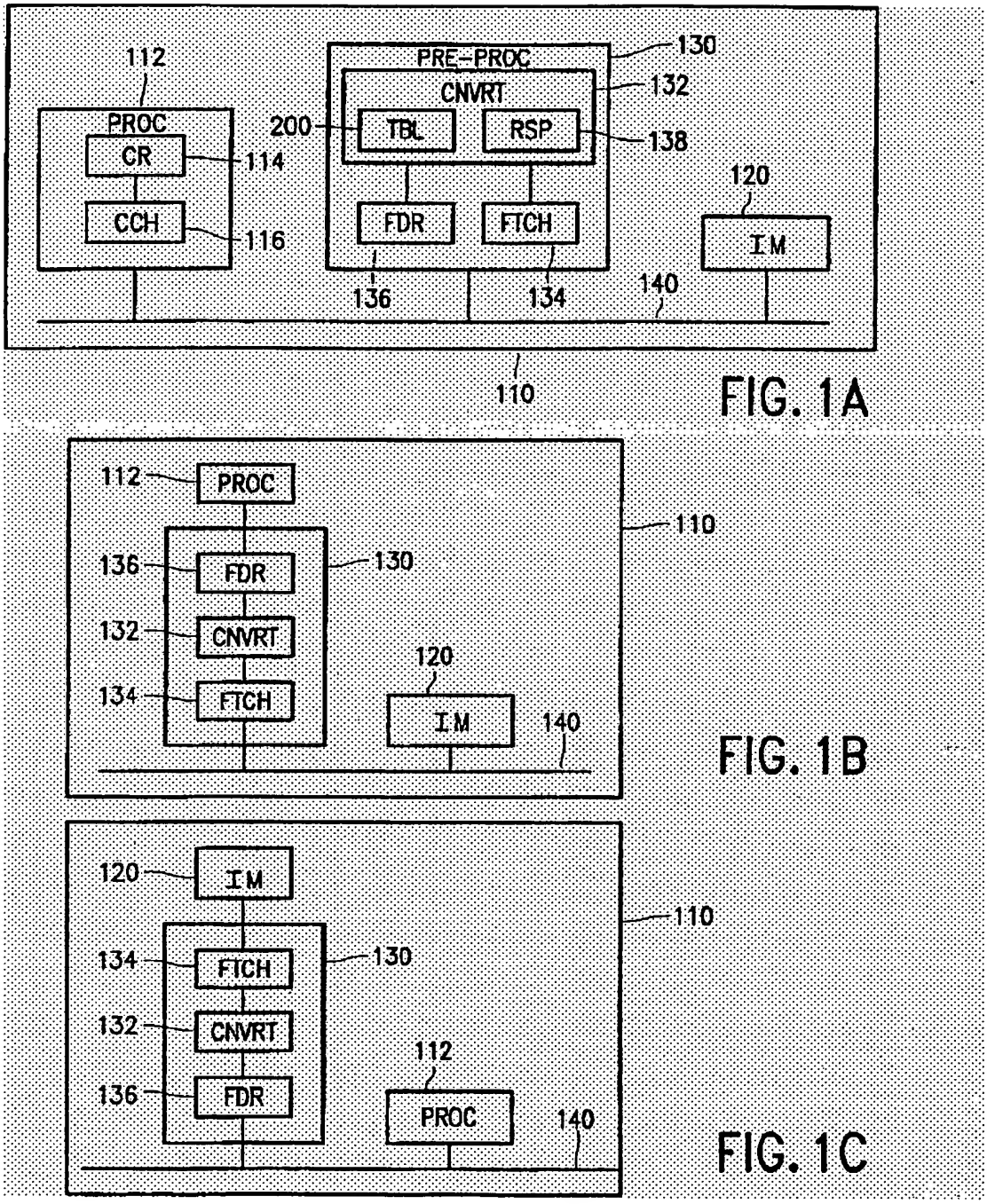

FIG. 1A

FIG. 1B

FIG. 1C

【図１】

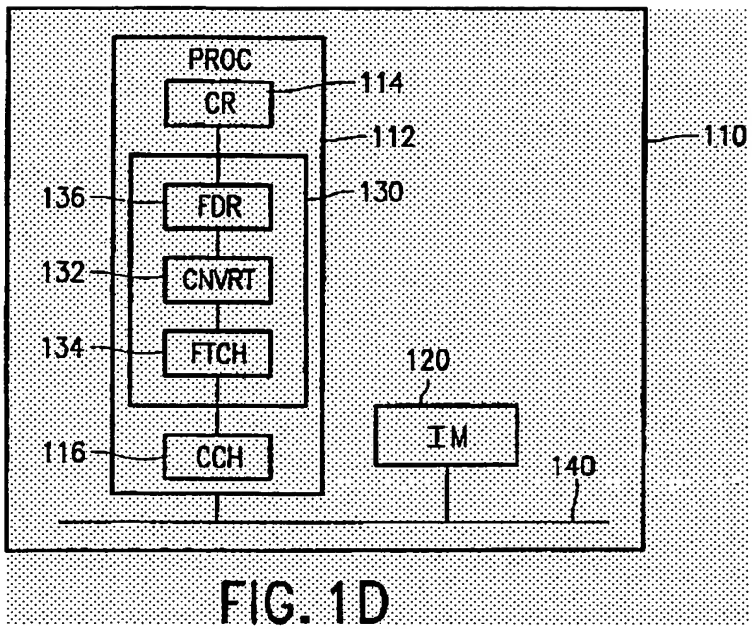

FIG. 1D

【図２】

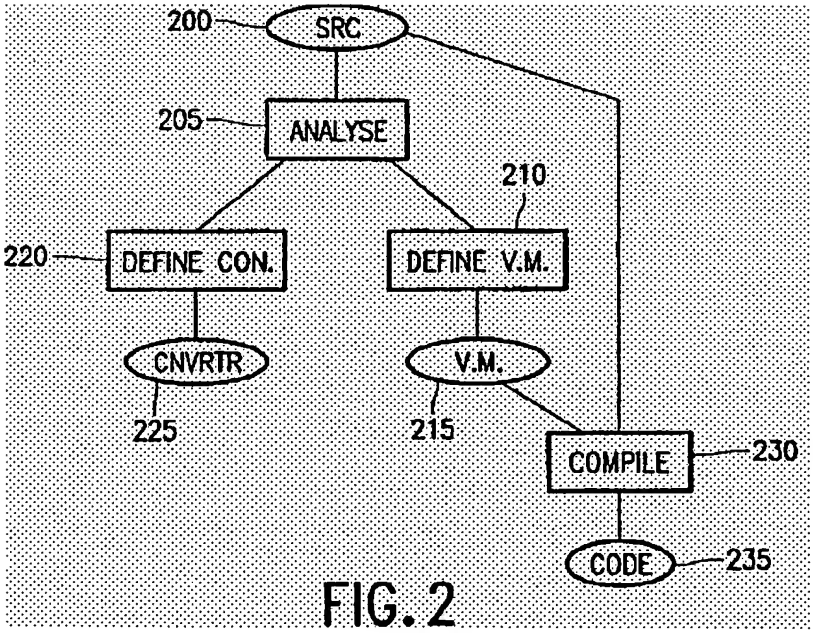

FIG. 2

【図３】

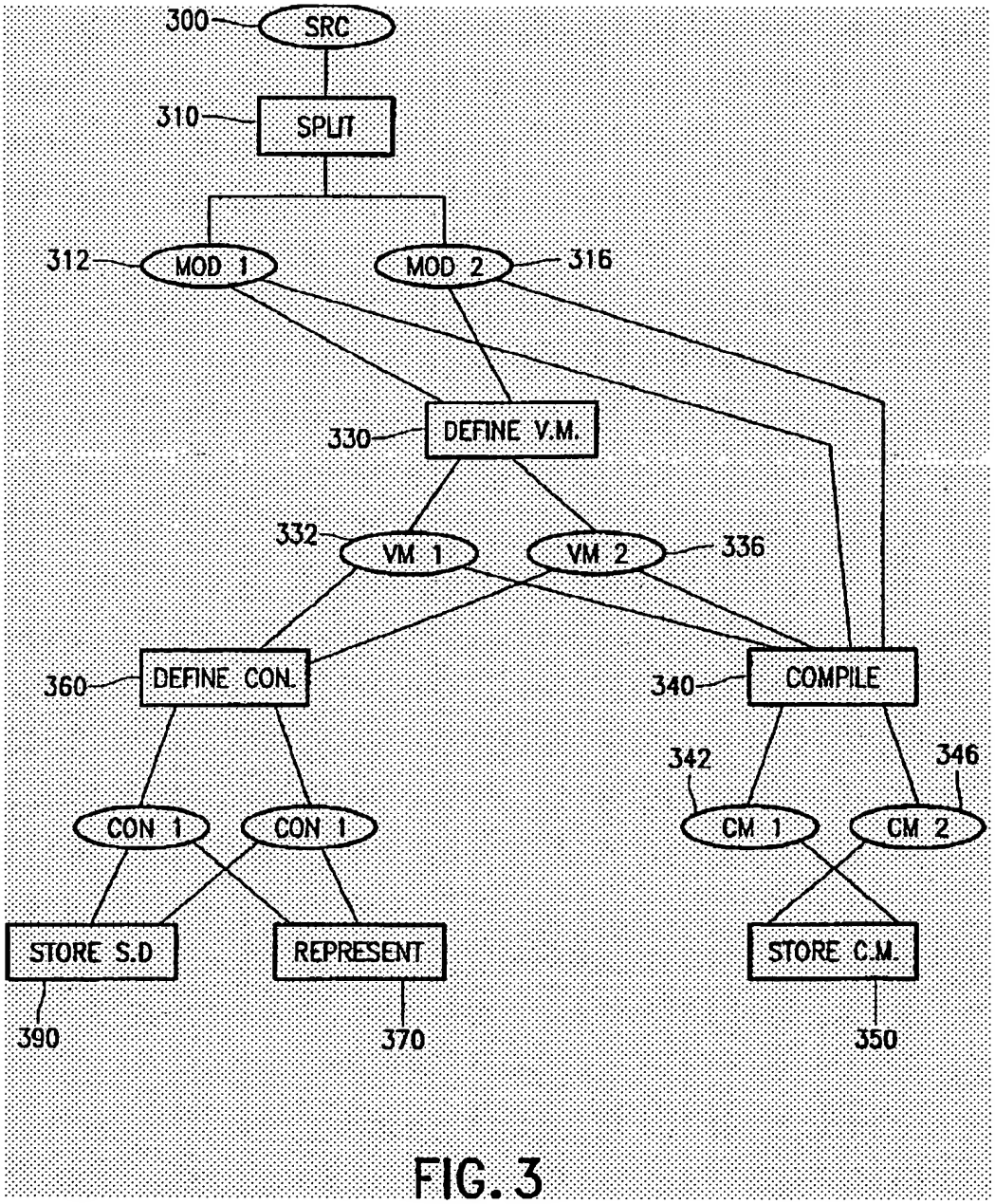
300
SRC
310
SPLIT
312
MOD 1
MOD 2
316
330
DEFINE V.M.
332
VM 1
VM 2
336
360
DEFINE CON.
340
COMPILE
342
346
CON 1
CON 1
CM 1
CM 2
STORE S.D
REPRESENT
STORE C.M.
390
370
350

FIG. 3

【図４】

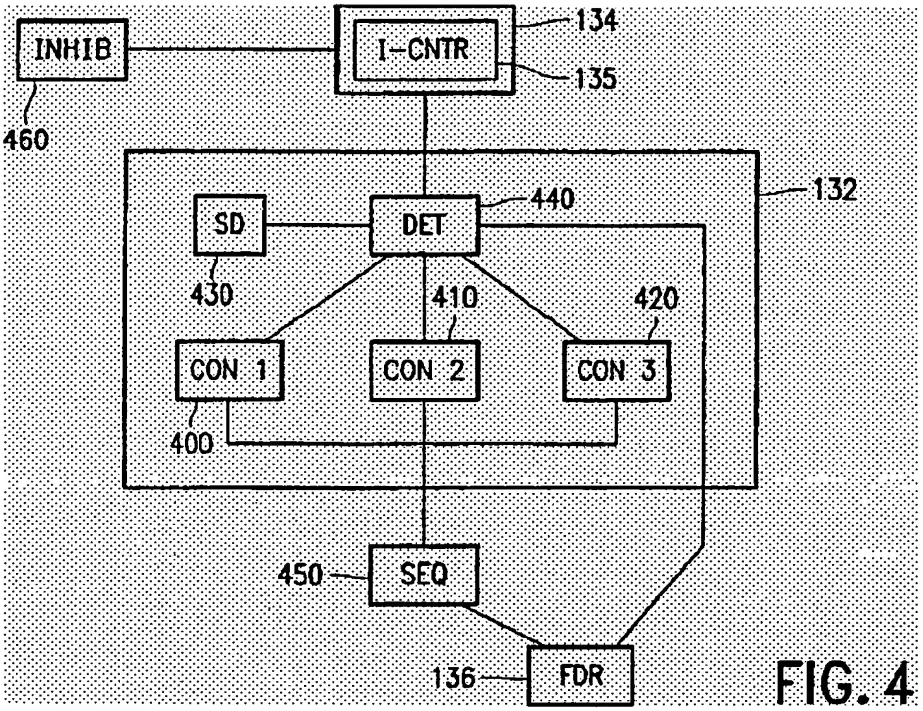

FIG. 4

【図５】

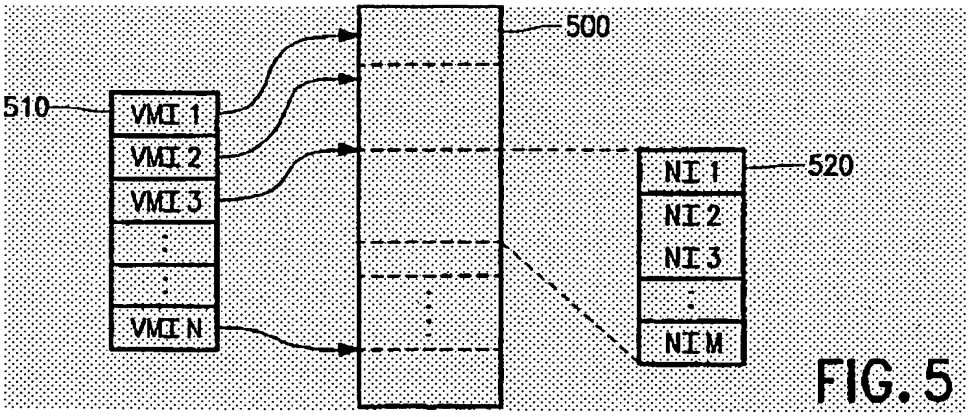

FIG. 5

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB 98/01453

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: G06F 9/318, G06F 12/08 // G06F 17/30
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC6: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

DIALOG, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 9727537 A2 (SUN MICROSYSTEMS, INC.), 31 July 1997 (31.07.97), page 4 - page 5, figure 6A, claim 1,6,12-13,19,23	1-2,4-6
A		3,7-12
	--	
Y	WO 9723823 A2 (PHILIPS ELECTRONICS N.V.), 3 July 1997 (03.07.97), see whole document	1-2,4-6
A		3,7-12
	--	
A	US 5586323 A (SHINOBU KOIZUMI ET AL), 17 December 1996 (17.12.96), column 4, line 1 - column 6, line 65, figures 3,5l, claims 1-3, abstract	1-12
	--	

[X] Further documents are listed in the continuation of Box C.　[X] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 April 1999	22-04-1999
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. + 46 8 666 02 86	Authorized officer Linus Wretblad Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT	International application No. PCT/IB 98/01453

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Department of Electrical Engineering and Computer Sciences University of California, Berkeley, 94720 The convergence of telecommunications and computing: What are the implications today? David G. Messerschmitt IEEE Proceedings, August 1996 see pages 17-22 —	1-12
A	WO 9727536 A1 (SUN MICROSYSTEMS, INC.), 31 July 1997 (31.07.97), page 3 - page 4, claims 1,8-10, abstract —	1-12

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

02/03/99

International application No.
PCT/IB 98/01453

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9727537 A2	31/07/97	WO 9727536 A	31/07/97
		WO 9727539 A	31/07/97
		WO 9727544 A	31/07/97
WO 9723823 A2	03/07/97	EP 0811190 A	10/12/97
		GB 9526129 D	00/00/00
		US 5872978 A	16/02/99
US 5586323 A	17/12/96	EP 0510616 A	28/10/92
		EP 0834803 A	08/04/98
		EP 0838754 A	29/04/98
		JP 4322329 A	12/11/92
WO 9727536 A1	31/07/97	WO 9727537 A	31/07/97
		WO 9727539 A	31/07/97
		WO 9727544 A	31/07/97

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(81)指定国　　EP(AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), JP
(72)発明者　ホリッセン　ポーラス　エム　ハー　エム　アー
オランダ国　5656　アーアー　アインドーフェン　プロフ　ホルストラーン　6
(72)発明者　メレンブルークス　フランシスカス　イェー　ハー　エム
オランダ国　5656　アーアー　アインドーフェン　プロフ　ホルストラーン　6
(72)発明者　ストラフェルス　ポール
オランダ国　5656　アーアー　アインドーフェン　プロフ　ホルストラーン　6
(72)発明者　トレスヘル　ヨーアヒム　アー
オランダ国　5656　アーアー　アインドーフェン　プロフ　ホルストラーン　6
【要約の続き】
令を、コア（114）により実行されるネイティブ命令に変換するために使用される。